Using VSAM with Com-plete

This chapter covers the following topics:

- Introduction
- VSAM Record Sharing and Integrity Options
- Using VSAM Files Online

Introduction

VSAM is an IBM access method that provides three file types, secondary indexing capabilities, and a useful utility referred to as Access Method Services (better known as IDCAMS).

The three file types, or clusters in VSAM terms, are:

- KSDS Key Sequenced Data Sets;
- ESDS Entry Sequenced Data Sets;
- RRDS Relative Record Data Sets.

The Key Sequential data sets are indexed data sets whose key is some variable or combination of variables in a file. The file is composed of two parts (actually two subfiles):

- Index component;
- Data component.

The index component contains the keys and a pointer to the associated data record. This pointer is often referred to as an RBA or Relative Byte Address. The data component contains the data record associated with the key. Note that the primary key must be unique within the file; however, any secondary keys (or alternate indices in VSAM terms) may have duplicates. KSDSs can be addressed either sequentially, randomly, or skip sequentially.

The ESDSs are sequential files; they may be addressed randomly through a Relative Byte Address (RBA) from the beginning of the file.

The key for a RRDS is the Relative Record Number, much like an Adabas ISN. In other words, if you want the third record in the file, the key would be three.

All VSAM files are divided up into logical units referred to as control intervals and control areas. A control interval consists of what could be considered a blocked record, i.e., one with multiple records per control interval. A control area is nothing more than a block of control intervals. Records may be variable or fixed lengths and may span control intervals.

VSAM has a much more efficient record storage concept for its KSDS than for other indexed access methods. It automatically builds multiple levels for its index components, and also handles new insertions, which cause other records to be shifted in an efficient manner.

The buffering services and sharing of control blocks are of great use in a multi-region/environment.

One of the capabilities of VSAM is its record sharing capabilities, which must be taken into consideration when using VSAM in an online environment. These capabilities and restrictions are discussed below.

From the application standpoint, the use of VSAM with Com-plete is a very easy task. The application programmer writes standard OS/VS COBOL VSAM I/O statements to access any VSAM files - just like writing batch programs using COBOL. The only special processing the application programmer must be concerned with is the checking of VSAM status codes and the writing of Com-plete screen I/O calls.

Important:

The file status codes must be checked after every VSAM I/O. For instance, if you do not check the VSAM status code after an OPEN statement and a bad open was encountered, you will get a "0C4" completion code when you try to reference the I/O area. This completion code indicates that addressability to the I/O area is not established until a valid open has been accomplished.

VSAM can provide data integrity at the control interval level (that is, VSAM reserves a control interval for one user and does not let any other users access this control interval until the first user releases this control interval). The types of integrity that VSAM provides are based on the share options specified at the time the clusters are defined. If a record is currently held by a user, a "93" status code is returned and the I/O call should be reissued. VSAM does not queue I/O calls that cannot be serviced. In other words, if a record is held by another task, VSAM will tell you about it, but it will not wait for that record to be freed before returning to you.

Normally, the procedure would be to issue a Com-plete rollout function (i.e., wait) and then reissue the call. If after several tries (a maximum of 10 times) the record is still not free, return a message to the screen indicating that the record is currently held and an attempt should be made later.

As an alternative, you can request Com-plete to serialize I/O requests for a file, thus avoiding a record not being available due to its being held by another user of the same Com-plete. For details, please refer to the description of the UUTIL online utility in the Com-plete Utilities documentation.The status codes and reasons that are most often returned are listed in the following table:

Status Code	Reason	Action
90	Bad OPEN: File was not closed properly by a previous user.	Run an IDCAMS VERIFY
93	Record or resource is not available Either another user has the record or the file is being used by another task in another region. On an OPEN, there is not enough virtual storage in Com-plete’s region.	Ensure no other job is using the file. Use the FM function of UUTIL to check if the file is open online. Ascertain if it may be useful to request I/O serialization by Com-plete.
95	Logic error in I/O call. Usually your FD in COBOL coes not match that defined in the VSAM cluster definition.	
96	The file is not defined correctly in Com-plete.	Use the FM function of UUTIL to define the file.

The *COBOL Programmers's Guide* provides a more detailed description of VSAM status codes and should be consulted. In addition, be aware that COBOL status codes have a different meaning based on the request type, that is, Open, Read, Write, etc. The *COBOL Programmer's Guide* also contains a table indicating the associated VSAM error codes and includes definitions of the codes.

There are several considerations to be made in cataloging a VSAM "DDN" to Com-plete. They include:

- Locally shared vs. non-shared resources;
- COBOL use of VSAM string processing;
- Defining buffer areas of VSAM files;
- Defining share options in the VSAM cluster definition.

VSAM Record Sharing and Integrity Options

Overview

The VSAM record sharing options are defined in the SHARE OPTIONS parameter when defining a cluster. Although VSAM allows sharing options for both across-region and across-system, only across-region options are discussed below. Com-plete users in different threads fall into the across-region category.

The share option "1,3" provides Com-plete read/write data integrity, that is, multiple users can access a data set for read processing while another user has it for update processing. Under this option, a "read" user cannot process a control interval that is currently being updated (read integrity). No two users can access the same control interval for update, thus providing write integrity; however, they can simultaneously update data in different control intervals.

The share option "2,3" is very similar to the "1,3" option, but it does not provide read integrity. In other words, it is possible for a read user to access the same control interval that another user is updating, thus losing read integrity. There is no problem in updating with this option because VSAM ensures that only one user has a control interval for update.

Note:
If you specify options "2,3" but access the data in a shared resources environment (that is, the Com-plete DDN is cataloged with LSR instead of NSR), VSAM forces share options "1,3". So if you want a "2,3" option, you must catalog the DDN with NSR, non-shared resources.

You are recommended to use VSAM option "2,3" if you want to read a file in a batch program while the file is still online.

There are two other options: "3,3" and "4,3". Neither one of these provide read or write integrity. It is up to you to provide this integrity (which requires Assembler level coding). You are not recommended to use these options unless it is absolutely necessary.

Note also that a control interval is not released from an update held status until the update user either accesses another control interval, issues a ENDREQ assembler macro, or closes the file. This can lead to problems where a record is read for update and sent to the screen. The control interval is tied up until the user finally responds, unless the program closed the file before returning to the screen. There are two options:

- Opening and closing a file on every scheduling of an update (never do this on a read-only program);
- Allowing the records in a control interval to be tied up until the user responds.

If you choose to use the second approach, inform the user that the records were held and that an attempt should be made later.

Buffers

Another important factor in the performace of VSAM processing is the use of buffers. VSAM automatically allocates one index buffer and two data component buffers for each string of a KSDS. (A string is a unique access path to a portion of a file.) Each string requires that VSAM maintain the position in the file. The amount of space required for a buffer is determined by the control interval size. So the larger the control interval size, the larger the buffer space. This also means that you have more records available. If the user is using sequential processing, then performance can be increased by using more data component buffers. If a user is using direct processing, then performance can be increased by using more index buffers. More index buffers are required when direct processing is used because VSAM accesses the index component for every request and refreshes the data component on every request. Thus, the more index control intervals kept in memory, the fewer I/Os needed to access records. Under sequential processing, VSAM does not access the whole index on every request but will instead use the horizontal pointers in the sequence set (the lowest level of an index) to access the next record. VSAM can use additional data buffers to do read-ahead functions and thus have data available for future requests. Note that sequential processing does not refresh its data component buffers on every request.

A complete discussion on optimizing performance and VSAM buffering techniques is available in the *VSAM Programmer's Guide* and should be consulted. This information is extremely important when you define the size and number of buffers to allocate when using the LSR (Locally Shared Resources) option. One difficulty encountered in using COBOL, Com-plete, and VSAM is the consistency and meaning of terminologies used in the different documentations involved. The following information should aid you in resolving this problem.

- COBOL does not support skip sequential processing, so there is no need to specify the SKP option when cataloging a DDN to Com-plete.
- The way COBOL determines whether you are going to update, or simply read a file is based on how a file is opened, that is, I/O=update, INPUT=read-only, OUTPUT=update.
- COBOL does not support addressed processing, but does support key processing.
- COBOL does not support multiple string processing. However, multiple strings to a data set are acceptable because of the concurrent running of programs accessing the same file from different threads.
- There is one RPL (Request Parameter List) for sequential processing and a second RPL for direct processing.
- The definitions of the option codes you specify when cataloging a VSAM DDN to Com-plete are explained in the VSAM Programmer's Guide.

- You must consult the following documentations when using VSAM, COBOL, and Com-plete:
    - Access Method Services
    - VSAM Programmer's Guide
    - VSAM Programmer's Guide for Advanced Applications
    - OS/VS COBOL Compiler and Library Programmer's Guide
    - OS/VS COBOL documentation
    - Com-plete Application Programming documentation
    - Com-plete System Programming documentation
    - Com-plete Utilities documentation (particularly ULIB)
- The following table relates some COBOL terms to VSAM terms.

COBOL Term	Associated COBOL Verbs/Clauses	VSAM Term
Random processing	ACCESS IS RANDOM or ACCESS IS DYNAMIC, READ, WRITE, REWRITE, DELETE	Direct processing
Sequential processing	ACCESS IS SEQUENTIAL or ACCESS IS DYNAMIC, READ NEXT, START, WRITE, REWRITE, DELETE	Sequential processing
File status code	FILE STATUS IS (COBOL interprets the VSAM codes and returns its own codes)	Return Code.
Current record pointer	N/A	Position within file
READ	READ	GET macro
WRITE	WRITE	PUT macro
REWRITE	REWRITE	PUT macro with update
DELETE	DELETE	ERASE macro

Using VSAM Files Online

The following sections contains notes on using VSAM files online.

- VSAM Files
- Alternate Indices

VSAM Files

1. If the file is to be updated, use MACR = (..NDF..). This slows down performance but ensures data integrity.

2. Make any file known to Com-plete reusable unless you are going to define an alternate for it. But remember that whenever that file is opened for output, the file is automatically emptied.

3. Make the data CI size small (1024,2048,4096), index CI = 512.

4. Avoid CA and CI splits by using FREESPACE for a volatile file.

5. Use spanned records with variable length records where only a few records are very large.

6. Avoid files where the records are deleted from the bottom and added to the top.

7. Do not specify ERASE unless the data is sensitive and all traces of it must be zeroed out.

8. STATUS-KEY = 96 (COBOL) and ONCODE = 1027 (PL/1) mean that the record specified is being held by another program that has the file opened as I/O.

9. ONCODE = 1028 (PL/1) means that after the file was opened by your program, someone else put the file in batch status and your program then tried to read the file.

10. Do not use IDCAMS to verify files owned by Com-plete while Com-plete is running. Set them to BTCH status first.

11. Use SHAREOPTIONS (2,3) when defining a cluster in order to avoid problems with one program trying to access a file as I/O while another program has that file accessed as input.

12. The application programmer is responsible for VSAM file backup and recovery. The best method to backup a file is to use a tape. The second best is to REPRO the file to another disk pack. It is not a good idea to back up a VSAM file to the same pack since a head crash or other disk pack error could wipe out both the real file and the backup file. It is also better to use REPRO instead of IMPORT/EXPORT to back up a file, because REPRO can reorganize the file at the same time (to reorganize the file, you must REPRO it back).

13. Use the appropriate VSAM prefix for that pack when naming a VSAM file (VSAM2. for MVS002). The second level name should be the valid prefix for that department. Should you need to change the name of any VSAM file, use the ALTER command of IDCAMS. Name the data and index components as well as the cluster.

14. Do not issue terminal I/O while holding any VSAM resources, for example, between GET for update and UPDATE, or in the middle of sequential VSAM processing.

15. Name both the data and index component as well as the cluster. This makes it much easier if you have to perform an ALTER on the VSAM file.

16. Thread-lock all programs using the same file(s) to the same thread, or request I/O serialization by Com-plete.

17. If you get an ONCODE = 92 when you open the VSAM file and the accompanying message says "DATA SET UNAVAILABLE", the file is probably offline (batch status). Another reason may be that the DDN is not defined to Com-plete.

18. If you get an ONCODE = 92 and the accompanying message says "DATA SET NOT PROPERLY CLOSED", it means that a job went down in the middle of VSAM processing and left the file open. To avoid this, include an IDCAMS verify step in the Com-plete initialization procedure, specifying all VSAM data sets that may be affected. Com-plete closes all VSAM files in use during termination processing, irrespective of whether it is terminating normally or due to an ABEND or CANCEL.

Alternate Indices

1. Use SHAREOPTIONS (2,3) for both the base cluster and the alternate index.

2. When using a generic key, you may want to add a high value record to the file to avoid end-of-file problems (particularly if you get an ONCODE = 1026).

3. A base cluster cannot be reusable.

4. The base cluster's DDN must be five characters long and each alternate index built upon that base cluster must use those same five characters plus a single digit.

5. In both the base cluster's DDN definition and the alternate index's DDN definition, use MACR=(..NSR..).

6. Make sure you define the path to Com-plete, not the alternate index. If you read the file using the path, the data is put in the base cluster. If you read the file using the alternate index, however, all the keys of the base cluster contain that alternate index key.

7. The recordsize (RECSZ) of the alternate index cluster is determined by the number of non-unique keys expected. The data consists of five bytes used by VSAM, the alternate key, and all of the base cluster keys that have this alternate key. Watch for any one alternate key having a large number of non-unique occurrences (the alternate KEY = blanks or zeros, etc.); this determines the maximum value of the record size.

8. In building the alternate index, IDCAMS performs an internal sort. If you have a large number of records in the base cluster, you must give IDCAMS a larger region and some additional JCL in order to build the indexes.

```
 //BUILDALT JOB (990030,1,1,0),TIM,CLASS=C
//BUILD    EXEC PGM=IDCAMS,REGION=600K
//SYSPRINT DD  SYSOUT=A
//STEPCAT  DD  DISP=SHR,DSN=CATALOG.VIPORES
//IDCUT1   DD  DISP=OLD,AMP='AMORG',VOL=SER=IPORES,UNIT=3375
//IDCUT2   DD  DISP=OLD,AMP='AMORG',VOL=SER=IPORES,UNIT=3375
//SYSIN    DD  *
BLDINDEX
        INDATASET(VSAMIPOR.SYS1.TESTINDX)
        OUTDATASET(VSAMIPOR.SYS1.TESTALT)
        CATALOG(CATALOG.VIPORES/PWUPDATE)
```

9. Whenever you delete the base cluster, IDCAMS automatically deletes any and all paths to that base cluster.

10. When using the alternate index to look at the data, changing the alternate key by means of the base cluster, and then trying to look at the old alternate index key again, watch for this potential problem: VSAM looks at the data in the buffers (in this case the alternate index) first and use it without actually doing another read of the file. Your program then tries to read the file using the old alternate key and fails because that record is no longer there. The solution is to flush the buffers. One way is to have a high value record in your file and read it by means of the alternate key which will change the buffers of the alternate index. (Note that the above problem will give you an ONCODE = 1030.)

11. Freespace can also be defined for an alternate index and should be used if any records are to be added to the base cluster. CI and CA splits seem to cause problems for alternate indices.

12. When the alternate index is built (by IDCAMS), the records are in base cluster key order. However, any adds to the file after the base cluster is built are added to the end of the alternate index and are not in base cluster key order.

13. In Pl/1, there is a built-in function called "SAMEKEY", which can be used if a VSAM file is being accessed by an alternate index that has non-unique keys. If any further records exist with the same key, it returns a "1"B.

14. Under Pl/1, it is possible to define one alternate index path as forward and another path as backward. This would allow the user to not only page forward from any given key but also to page backward. One constraint is that a file declared as backward can not also be declared as GENKEY. You are therefore required to use the full key for the starting point of the backward read.

15. If you try to read past the end of the file using the alternate index, an ONCODE = 1030 is issued.